Implementation of conceptual product models into configurators: From months to minutes

Haug, Anders; Hvam, Lars; Mortensen, Niels Henrik

Published in:
Proceedings of MCPC 2009

Publication date:
2009

Document Version
Publisher's PDF, also known as Version of record

Implementation of conceptual product models into configurators: From months to minutes

Anders Haug
University of Southern Denmark
adg@sam.sdu.dk

Lars Hvam
Technical University of Denmark
lhv@man.dtu.dk

Niels Henrik Mortensen
Technical University of Denmark
nhmo@man.dtu.dk

Susanne Lundvald
Novenco
slk@novencogroup.com

Peter Holt
Novenco
pth@novencogroup.com

Abstract. For years the use of software-based product configurators has produced a number of benefits for engineering-oriented companies. However, achieving such benefits can be challenging, and often configurator projects do not succeed. A main reason for such failures is that the tasks of developing and maintaining configurators often are very challenging and time-consuming. With a focus on reducing the efforts needed for development and maintenance of product configurators, this paper describes an emerging technology that makes it possible to automate the conversion of conceptual product models made by ordinary product experts into the knowledge base of a configurator, and the other way around. Thus, this new technology enables new ways of carrying out the tasks of configurator development and maintenance. This paper defines the new use patterns that the technology enables and deduces the possible benefits compared to existing approaches. To investigate if the new technology can fulfil its great promises, a case study is presented in which the technology has been applied.

Keywords. Product configuration, Product configurator, Configurator development, Configurator documentation, Conceptual modelling.

1 Introduction

A product configurator is a software-based expert system that supports the users in the specification of customized products by restricting how different elements and their properties may be combined. The use of configurator technology means that product specification tasks, which normally require human experts, can be automated. This paper focuses on engineering-oriented companies, which represent the most complex cases of configurator technology. In engineering-oriented companies configurators have been reported to provide a number of benefits, such as: shorter lead times, improved quality of product specifications, preservation of knowledge, use of fewer resources for specifying products, optimized products, less routine work, improved certainty of delivery, and less time needed for training new employees (e.g. Ardissono et al., 2003; Hvam, 2004; Forza and Salvador, 2002; Forza and Salvador, 2007). However, achieving such benefits is by no means a guarantee when initiating a configurator development project, but in relation to the development and the maintenance of a configurator, there are great challenges to face.

Gathering and representing relevant information is one of the most difficult tasks in configurator projects (Sabin and Weigel, 1998; Hansen, 2003; Hvam et al., 2008). Thus, in many cases, before initiating the development of the configurator, the information to be included is firstly defined in conceptual models to allow a clear focus on product information, while, to some extent, avoiding consideration of how this should be implemented later. Conceptual models provide a basis for discussing what to include in the configurator and how this information is structured. Thus, relevant product experts need to understand the conceptual models. Since product experts most often do not posses expertise in conceptual modelling or software development, the language used for the creation of conceptual models needs to be relatively easy to understand. Upon completion of the conceptual models, these are to be implemented into the knowledge base of a configurator, which are manual tasks of translating conceptual models to the configurator software syntax and typing this into the configurator. When a configurator has been developed and implemented in an organisation, the maintenance phase begins. In this phase, the configurator needs to be updated when the products in focus change. In many cases such updates imply a need for product experts to evaluate the existing implemented knowledge in order to change or extend it. However, the modelling environment of configurators seldom provides overviews of the implemented product information that are adequately comprehensible for the product experts. Thus, for product experts to understand what is implemented in the configurator, external representations are often needed.

Documentation of configurator knowledge bases is, however, a time-consuming task that often seems to be neglected (Edwards et al, 2005; Hvam et al., 2005).

This paper focuses on making the tasks of developing and maintaining complex product configurators easier. Thus, the paper describes and evaluates a new type of software module, which aims at automating the information transfers from the conceptual models to configurator software and the other way around. Obviously, an automation of such processes promises large reductions of the efforts needed for the development and maintenance of configurators. However, it needs to be clarified what this software module actually offers. Thus, this paper aims to answer the questions: (1) how can this new software module change the ways in which configurators are developed and maintained? and (2) how well does it work in practice?

The remainder of the paper is structured as follows. Firstly, section 2 describes how product information can be represented and resumes literature on documentation in configurator projects. Next, section 3 describes the methodology used to answer the questions in focus. Section 4 analyses a new technology for exchange of information between conceptual models and configurator software, and deduces the use patterns and possible benefits that the technology enables. Section 5 presents a case study in which the new technology has been applied. The paper ends with conclusions in section 6.

2 Background

This section shortly clarifies the type of documentation in focus, i.e. in which format the information can be. This is followed by a review of research done on the topic of documentation in product configuration projects.

2.1 Representation techniques for conceptual modelling

Normally, bills of materials (BOMs) are represented as flat or hierarchical lists of components with information such as item number, name, and quantity. Thus, representation of normal BOMs does not require special notation or software. However, representing an entire solution space for a product family (in a generic product information model), which includes hundreds, thousands or millions of possible instances (BOMs), posses other requirements on the representation formalism and the software used for creating and maintaining it. Three commonly applied techniques in configuration projects are: (1) the PVM technique; (2) class diagrams; and (3) CRC-cards (class responsibility collaborator) (Felfernig et al., 2000; Hvam, 2004; Edwards et al., 2005; Haug and Hvam, 2007a; Hvam et al., 2008).

The PVM technique is a formal way of representing generic product information. PVMs describe classes (representing components, assemblies or principle classes), class attributes (properties), relationships between classes, and constraints that define how classes and properties may be combined. PVMs consist of two generic sections, in the left side part-of (whole-part) structure and in the right side kind-of (generalization-specialization) structure. Different definitions of the PVM notation have been proposed by Mortensen et al. (2000), Harlou (2006) and Haug (2007). Based on the latter definition, which represents the most extensive and formalized of these definitions, a principal example of the PVM technique is seen in Figure 1. In the example, the guillemets (<<...>>) are used for indicating that the class 'BodyAssembly' is a class of the type 'assembly'.

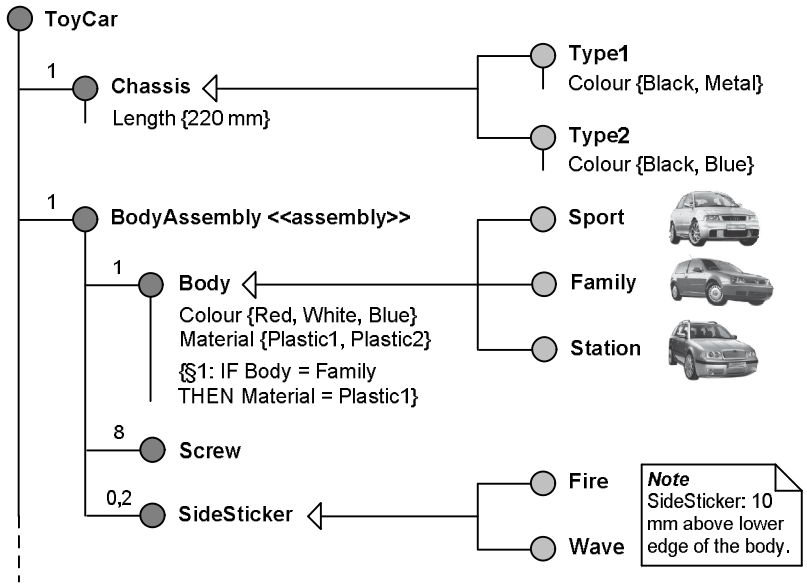

Figure 1. Example of the PVM technique.

Compared to PVMs, class diagrams are much more widely applied, not least in general software development. Class diagrams are part of the Unified Modelling Language (UML), which, in the 2.0 version, includes 13 diagram types (OMG, 2005) of which the class diagram is the most frequently used (Fowler, 2005). Figure 2 shows a principal example of a class diagram applied for describing product information. Here it should be noted that navigability arrows are sometimes added on the composition and association relationship lines.

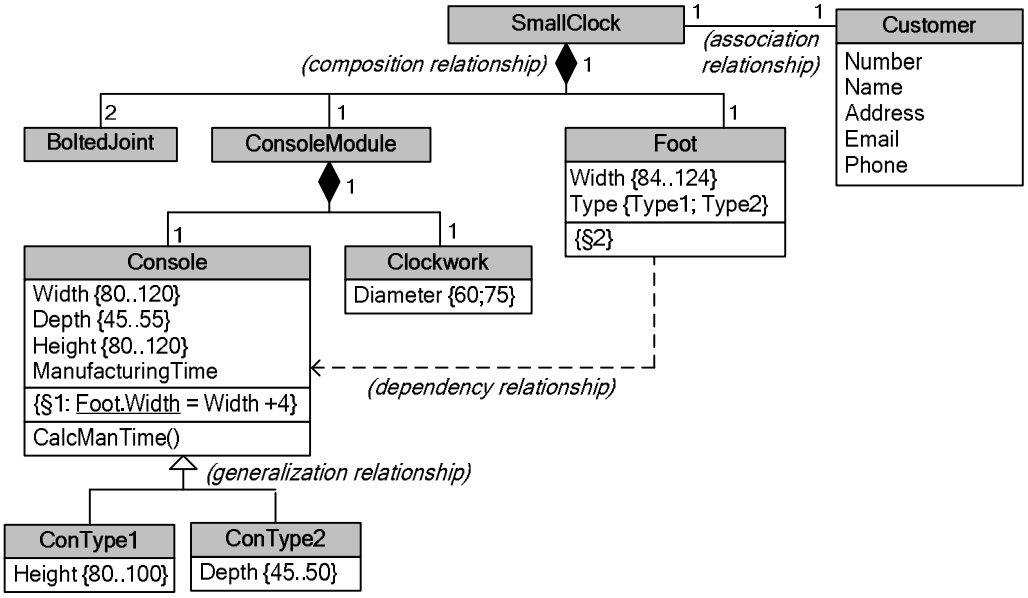

Figure 2. Example of class diagram.

The choice of using PVMs or class diagrams in configuration projects implies different advantages and drawbacks. Studies show that while class diagrams represent a richer, more flexible and more formal notation, PVMs are seemingly easier to learn and review for persons without experience in conceptual modelling or software development (Haug and Hvam, 2007a; Haug and Hvam, 2007b). Also, studies of Danish companies engaged in configuration projects show that PVMs (or variants of this notation) are much more commonly applied than class diagrams in such projects (Edwards et al., 2005; Hvam, 2004; Haug and Hvam, 2007a).

When using PVMs or class diagrams for the representation of generic product information, the amount of information to be included in such models can become very extensive, implying that the overview of these diagrammatic models is lost. To avoid this, sometimes detailed information about classes are placed in CRC-cards (Class-Responsibility-Collaboration-cards) (Hvam et al., 2008; Haug and Hvam, 2009). CRC-cards include a number of different fields, which varies from company to company (Haug and Hvam, 2009). Figure 3 shows the CRC-card definition from the book by Hvam et al. (2008). In these cards the 'methods fields' are for describing information about methods and what is referred to in other contexts as constraints or rules. 'Product methods' concern products and their life-phase properties, while 'system methods' concern software aspects of a configurator. 'Internal methods' refer to the internal structure, functions and properties of a class, while 'external methods' refer to interfaces to others classes.

Class name:	Date:	Author/version:
Responsibilities:		
Aggregation	Generalisation	
Superparts:	Superclasses:	
Subparts:	Subclasses:	
Sketch:		
Attributes:	Collaborators:	
System methods:		
Product methods:		
Internal methods: External methods:		

Figure 3. Example of a CRC-card definition (Hvam et al. 2008).

2.2 Documentation of product information in configuration projects

During 2003 and 2004, the experience with product configurators in twelve Danish companies was investigated in a research project (Edwards et al., 2005). This research project showed that the documentation task was often one of the first areas to be given a low priority or even cut out of the project. The research project also pointed to a list of negative consequences of not documenting what had been implemented in the knowledge base of a configurator. For some of the investigated companies this meant that they were unable to further develop their configurators within reasonable use of resources, for which reason projects had been abandoned or configurators rebuild. Another unfortunate effect reported was that the lack of documentation had a negative impact on the daily communication between the people involved in product development and configurator development, because of misunderstandings and dysfunctional configurators.

In many cases, the lack of external documentation of what has been implemented in a configurator may be explained by a lack of appropriate documentation tools (Hvam et al., 2005). This is in particular the case when using PVMs and CRC-cards for which no dedicated software tools have existed until

recently, in contrast to tools for the creation of class diagrams. Thus, the software applied when creating PVMs and CRC-cards is such as MS Visio, Excel, Word, AutoCAD and Lotus Notes. However, since none of these programs are targeted at the creation of PVM or CRC-card models, they hold a long list of limitations. An example of such limitations is the missing link between attributes and the rules that use these attributes. This means that if an attribute in a PVM model is referred to in some rules, there is no mapping between the attribute and the rules, for which reason if the attribute changes, the rules become incorrect and must be changed manually. Also, if PVM classes are described in CRC-cards, there is no mapping between the descriptions in the PVM model and the CRC-cards, although they refer to or describe the same elements. Such issues can make the creation and maintenance of conceptual models very time-consuming and imply that errors occur. Responding to the apparent need for software better suited for the documentation of the configurator knowledge bases, research on this topic has been carried out.

Hvam and Malis (2001) investigate five standard configurator software shells. They conclude that none of these software shells include the needed features for supporting the documentation task in configuration projects. Based on their experience from several Danish configuration projects, Hvam and Malis (2001) define requirements for a documentation system: (1) easy to maintain, (2) facilitate the modelling techniques PVMs, class diagrams, and CRC-cards, (3) have central storage of data, (4) support network distribution of data, (5) support multiple user access, (6) integrate the modelling techniques, (7) include version control, and (8) allow integration with product configurators. Furthermore, they describe the development of a documentation system prototype at GEA Niro A/S, based on Lotus Notes.

Later, the prototype described by Hvam and Malis (2001) was further developed and taken into use by the companies GEA Niro A/S and American Power Conversion A/S (Hvam, 2004). The use of the Lotus Notes based documentation system has shown that there are great benefits from applying such a tool for the maintenance of product configurators, but also that the Lotus Notes based systems have significant limitations (Hvam, 2004). From a modelling point of view the Lotus Notes based documentation systems fail to offer support for the elaboration of class diagrams or PVMs, but only include CRC-cards together with a hierarchical list of classes that does not include attributes, constraints, or kind-of structure (generalisation). Consequently, both mentioned companies, which use the Lotus Notes application, apply other software for the creation of PVMs when making big changes or additions to their existing models. In Figure 4, a screenshot from the documentation system of GEA Niro is shown.

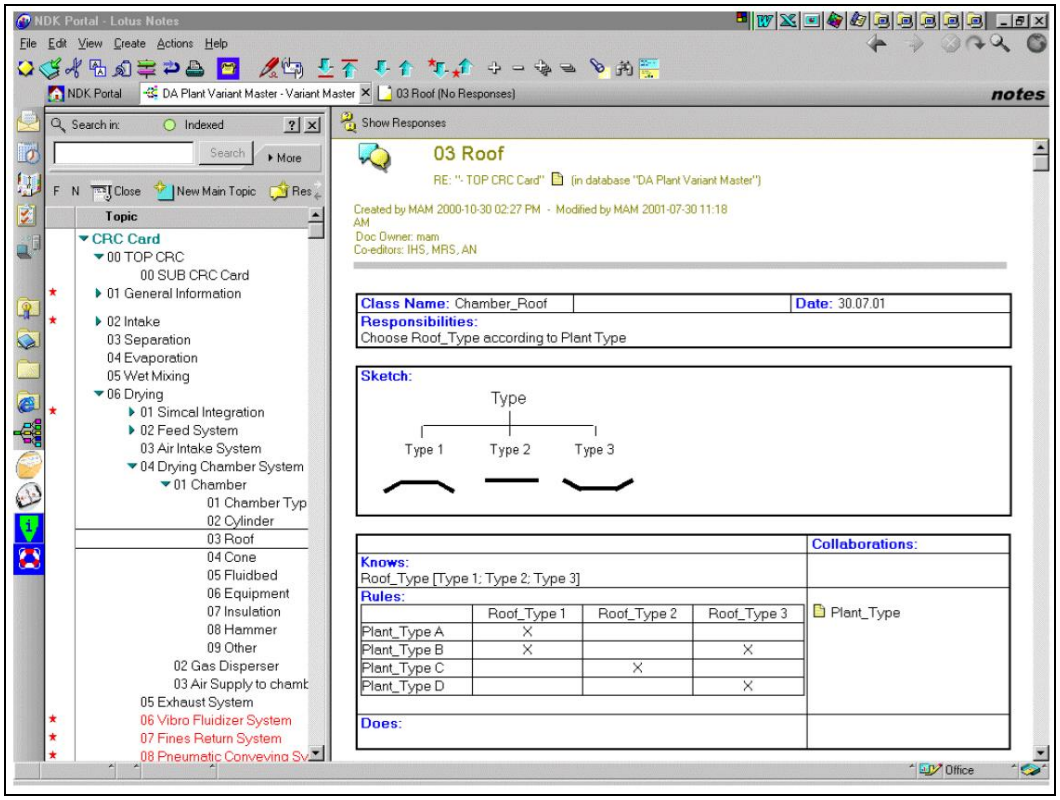

Figure 4. Lotus Notes a documentation system (Hvam and Malis, 2001).

Hvam et al. (2005) interviewed four Danish manufacturing companies that use product configurators in order to investigate what kind of documentation system support they need. They translate the answers into a set of overall requirements for a documentation system: (1) a coherent product model, (2) version control, (3) access control, (4) change notification, (5) user-friendliness, (6) web-based access, (7) integration to other software systems, (8) possibility of informal rule expressions, (9) hyperlinks to internal and external files, (10) flexibility, (11) configuration system integration, (12) the use of English as language, and (13) an inexpensive solution. Furthermore, Hvam et al. (2005) present a high-level description of a possible architecture for a documentation system.

Although Hvam and Malis (2001) and Hvam et al., (2005) mention which modelling techniques should be included in a configurator documentation system, this research does not in a detailed manner deal with topics like user interface design, detailed definitions of the included modelling techniques, and how to handle interrelated models (i.e. separate models that share elements). Thus, an important part of the basis for developing a documentation system was still missing. To address the need for adequately flexible and software-prepared notation formalisms of PVMs, class diagrams and CRC-cards in order to include these in a documentation system, Haug and Hvam (2007c) present such definitions. Another step towards the creation of a documentation system was

taken by Haug and Hvam (2009), who present a revised definition of the CRC-card layout, which takes into account future software-supported elaboration of the CRC-cards. This revised layout includes several new fields, e.g. for documenting additional relationship types and for organising information about attributes, constraints, and methods. Also, an extension of the CRC-card layout for inclusion of fields for managing change requests and versions is defined.

Based on the definitions by Haug and Hvam (2007c;2009) (produced in 2006), the first prototype to support the modelling techniques of the CPM-procedure in an integrated manner was created in 2006 (Haug et al., 2007). The main purpose of this prototype was to test the definitions of Haug and Hvam (2007c;2009). The tests included: (1) typing data from an ongoing configuration project into the prototype to investigate possible limitations of the modelling environment of the prototype, and (2) presenting the prototype for two companies, in order to compare the prototype with their current documentation software and to get other kinds of feedbacks. The tests to some extent confirmed that the defined design specifications were consistent and provided a solid basis for the creation of a configuration documentation system. However, because of the focus being on evaluating only certain aspects of the definitions and because of time-constraints, the prototype was not developed into a state in which it was applicable in practice. The most important limitations of the prototype are: it is too inflexible concerning the PVM and class diagram modelling; it does not support the generic CRC-card principle defined by Haug and Hvam (2009); it is not adequately stable or complete to be suitable for use in practice; and it includes almost no administrative functionality.

During 2007 additional specifications for a configurator documentation system were made by Haug and Lisbjerg (Haug et al., 2008). During this period, prototypes were continuously created and the existing design specifications were further developed. Based on the design specifications and the prototypes made, in the end a full-scale system was created. This software is named Product Model Manager (PMM) and the first version of this software was released in December 2007 by Incore Systems. PMM does not include a class diagram environment. This is based on the argument that the defined extensions of the PVMs notation imply that the combined advantages of both diagram types to some extent can be achieved. Thus, there is no need for supporting both diagram types (Haug et al., 2008). Besides the PVM view, PMM includes a Class Information Card view (an advanced form of CRC-cards) and a Product Model Explorer view. The Product Model Explorer view provides a simple overview of the elements included in a model and can be used for navigation between Class Information Cards and within a PVM model. Figure 5 shows the PVM and the Product Model Explorer views.

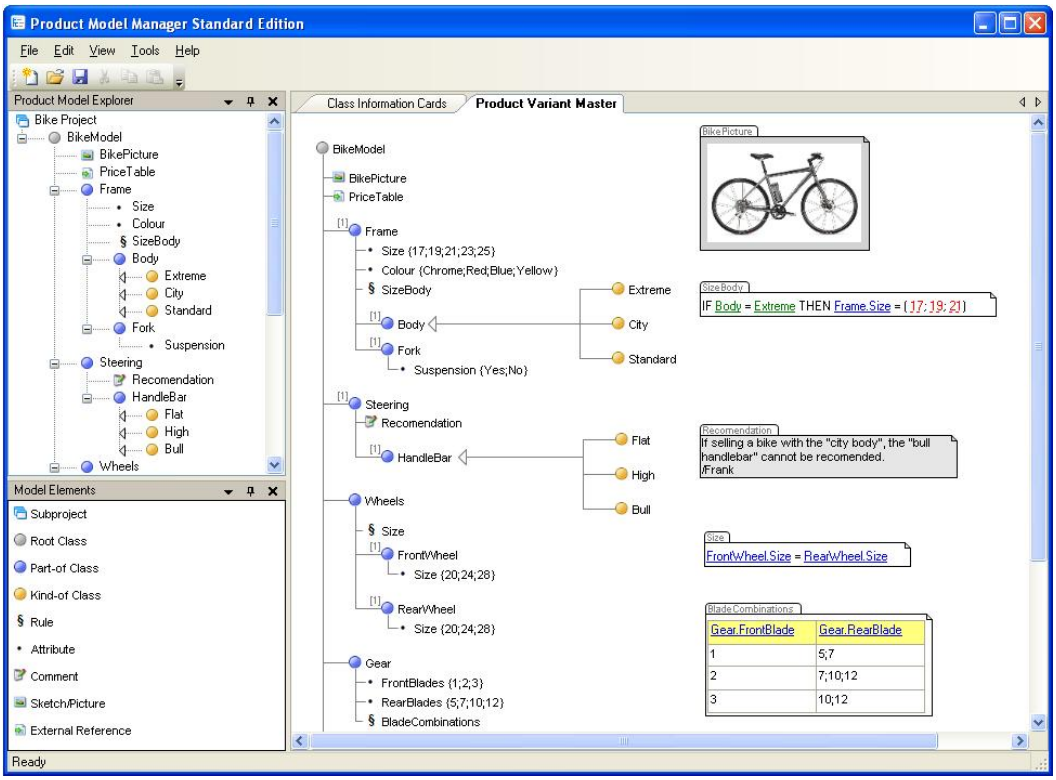

Figure 5. The PVM and Product Model Explorer environment (Haug et al., 2008).

Haug et al. (2008) compare PMM to the best existing alternatives for the creation of PVM and CRC-card models, i.e. MS Visio, MS Word and Lotus Notes. In the comparison between the use of PMM and MS Visio together with MS Word, 18 examples of PMM related advantages are mentioned. On the other hand, Haug et al. (2008) argue that the use of PMM for the creation of product information models holds no real drawbacks compared to the use of MS office tools. Haug et al. (2008) also compare PMM to the use of Lotus Notes for documentation of configurators and mention 18 examples of advantages that PMM holds in this context. Finally, Haug et al. (2008) argue that in a proper setup PMM does not have any real functional drawbacks compared to the use of Lotus Notes.

Haug et al. (2009) position PMM in relation to PDM (Product Data Management) systems and present a minor study of the use of PMM at a Danish Engineer-To-Order company. The persons interviewed in the case study answered that compared to using MS Visio and MS Word (which were used at the specific department before), the use of PMM reduce at least 50 percent of the time used on the drawing work when creating PVM models. The interviewed persons also were of the opinion that the modelling environments of standard configurator shells do not provide adequate user-friendly views of the knowledge base for product experts to understand what is implemented. Furthermore, the persons interviewed were not familiar with any PDM systems that provide the modelling features that

PMM does, and thus in many cases would offer inadequate support for the creation of generic product information models. Haug et al. (2009) conclude that although PMM bears some resemblance with PDM systems, PMM excludes some common PDM system features and has special features of its own, for which reason PMM may rather represents a new type of software. A main difference is that PDM systems focus on the management of data in static product structures, while PMM focuses on the modelling of generic product information, which often is found in contexts of engineered products.

3 Methodology

Two steps were carried out to investigate the two questions on which this paper focuses, i.e. how a new software module for integration of conceptual models and configurators can change the ways in which configurators are developed and maintained, and how well this software module works in practice.

The first step includes analysing the new software module for synchronization of models made in the conceptual modelling tool, PMM, and configurator software. From this point in the paper this new software module is referred to as the PMM-CSM (Product Model Manager - Configurator Synchronization Module). Based on the clarification of the functionality of the PMM-CSM, the paper deduces which new approaches for the development and maintenance of product configurators that the PMM-CSM enables. Finally, the defined new approaches are compared with traditional ways of carrying out configuration projects, and the possible benefits and drawbacks of the new approaches are analysed.

The second step focuses on investigating the use of the PMM-CSM in practice, in order to clarify the usefulness of it. This includes a study at a Danish company where PMM-CSM was applied. More specifically the study focused on: (1) clarifying the usefulness of PMM; (2) analysing the experience with the use of the new software module for transfers from configurators to PMM; and (3) analysing the experience with exporting models from PMM to configurators. Another use pattern was also to have been investigated, namely simultaneous update of models in PMM and a configurator. However, this approach had not been applied at this company at the time of the study, for which reason it is not included. The study of the use of PMM and the PMM-CSM was carried out as action research (Susman and Evered, 1978; Argyris et al., 1985), in the sense that the first author was the knowledge engineer in these projects. More specifically this included the task of assisting the acquisition of information from product experts, representing information in PMM, and making product experts evaluate the models created. Since action research implies two roles for the researcher, namely as a participant in the change process and as an observer that reports from the study, action

research, is somewhat in contrast to the positivist science tradition where researchers attempt to disengage themselves from their study subjects. Thus, in some parts of academia, action research has been criticised for being unscientific. On the other hand, since the areas typically investigated by using action research are often blurred (complex social systems), the intimacy of action research may be seen as a means of promoting appropriate change and understanding of practice (Waterman et al., 2001). The aim of this paper is to create reproducible results and knowledge without strict contextual limitations. Thus, in order to minimise the possible influence of bias and subjective evaluations, the case study was carried out under a set of research rules: (1) notes were taken by the researcher during the case; (2) experts from the company were interviewed in order to evaluate the description of the case; (3) all evaluations of resource/time savings were to be made by other persons than the researcher; and (4) the conclusions made based on the case studied focus on aspects with little relation to the way in which the study was carried out, i.e. a focus on which tasks that the technology could automate, rather than on how useful the technology is perceived to be.

4 A new approach for the development and maintenance of configurators

This section describes a new software module for synchronization of PMM models and configurators, and analyses which possibilities this new module provides in the context of developing and maintaining configurators.

4.1 The PMM-configurator synchronisation module (PMM-CSM)

As described in section 2, some good results had been achieved with the use of PMM for the creation of conceptual models of what to implement in a configurator and as documentation of what had been implemented. However, the ambition of automating the transfer of information from conceptual product models to the knowledge base of a configurator (Hvam and Malis, 2001; Hvam et al., 2005) had still not been fulfilled. To achieve this goal, an integration module for two particular types of configurator software shells was developed during 2008. The basic idea of this PMM-CSM is automation of transfers from conceptual models to knowledge bases of configurator software and the other way around.

The PMM-CSM supports five use patterns, as shown in Figure 6. Use patterns 3, 4 and 5 presume that a PMM model and a configurator model were previously synchronised, i.e. that the PMM model was created by exporting a configurator model or vice versa (use pattern 1 or 2).

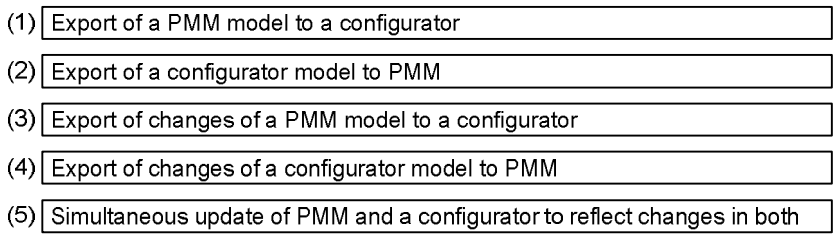

Figure 6. Use patterns supported by the PMM configurator integration module.

Figure 7 illustrates the principle of use patterns 1 and 3. The basic idea of this type of approach is that the product experts themselves (or assisted by a knowledge engineer) coordinate and formalise their product information into conceptual models. These conceptual models are then automatically transferred to the configurator software, in principle ready for compilation (possibly with some modifications made by a software expert).

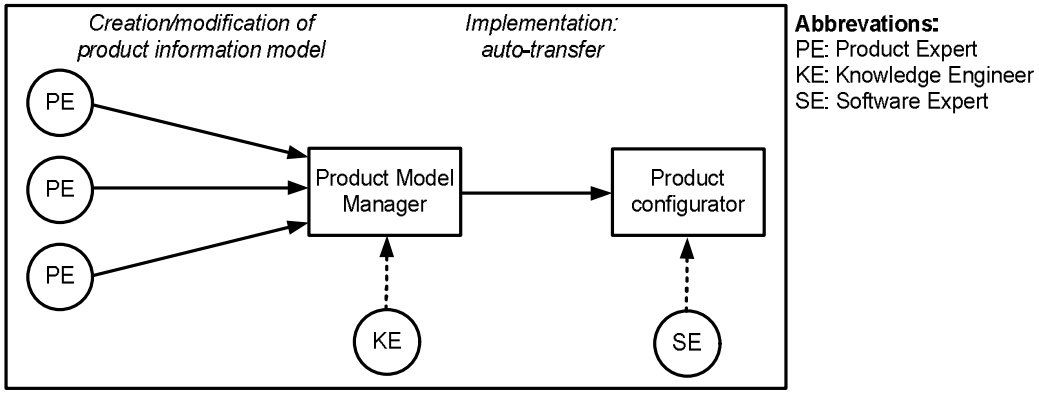

Figure 7. A new approach for configurator development.

Figure 8 shows use patterns 2 and 4. These use patterns are aimed at the documentation of already built configurator knowledge bases, which are translated into a language understood by common product experts.

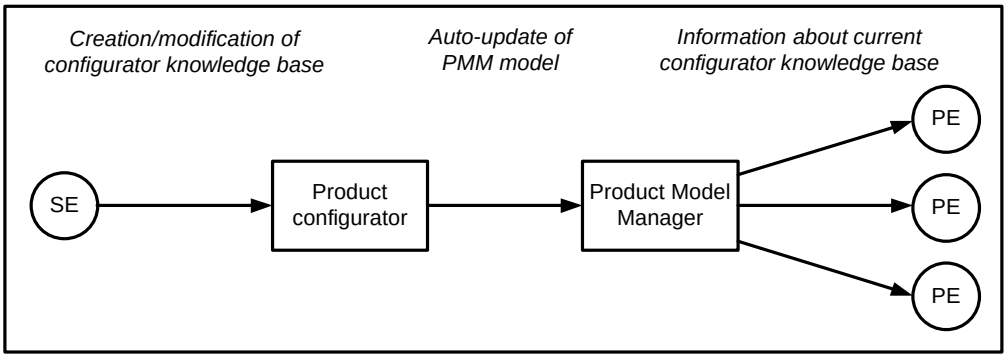

Figure 8. A new approach for configurator documentation.

Use pattern 5 can be perceived as a simultaneously execution of the processes described in Figure 7 and Figure 8. More specifically, this means that product experts can make changes of models in PMM, while software experts can make changes of the configurator, after which the changes of the PMM models are transferred to the configurator and the changes to the configurator knowledge base are transferred to the PMM model. If there is a conflict between two models during synchronisation, relevant persons can decide which model definition to respect.

4.2 Impacts on configurator development

To identify the possible benefits of the five use patterns defined in Figure 6, first a definition of the existing way of carrying out relevant tasks is required. To begin with, when developing product configurators, several basic processes need to be carried out by the ones given the responsibility of developing the configurator, i.e. knowledge engineers and/or software experts:

- Locating, retrieving and coordinating relevant product information
- Making conceptual models of product information (for overview and evaluation by product experts)
- Structuring product information according to the chosen configurator software
- Implementing information into the configurator software
- Facilitating tests of the configurator

For the first process, the difficulty of locating, retrieving and coordinating product information varies from project to project. What can make such tasks challenging is if the relevant information is spread across multiple product experts, information is poorly structured, some information does not exist in explicit form, or some product experts do not agree on what to include in the configurator. Process 2, the creation of conceptual models, is a phase which is sometimes skipped, and instead the information of product experts is implemented directly. The trade-off in this context is that while obviously it can be faster to skip process 2, in some cases there is a need for evaluation from product experts of what has been implemented. If it becomes troublesome to communicate with product experts because of the lack of external documentation, the resources needed for such communication can be many times greater than the resources saved by not creating external documentation. The last three processes require expertise in the configurator software chosen. Process 3-4, structuring product information according to the chosen configurator software and implementing information into the configurator software, are often challenging and time-consuming. Often there is not an unambiguous mapping between the information provided and the language of the

configurator software. Besides, the information delivered may be incomplete or self-contradicting, for which reason the persons creating the configurator have to collect new information and ensure that agreements are made. Process 5, facilitating tests of the configurator, implies that configurator developers ensure that product experts test the configurator, after which the configurator developers make the necessary modifications of the configurator.

The use of the PMM-CSM basically minimizes or even eliminates the need for knowledge engineers and/or software experts in the last three processes, i.e. structuring product information according to the chosen configurator software, implementing information into the configurator software, and facilitating tests of the configurator. But also in the first two phases the use of PMM and the PMM-CSM may imply some benefits. Table 1 describes such possible benefits.

Knowledge engineer/software expert task	**Possible benefits from using PMM and the PMM-CSM**
Locating, retrieving and coordinating relevant product information	Such tasks may be put more in the hands of the product experts themselves, rather than a knowledge engineer who may not know the product and the organisation as well
Making conceptual models of product information (for overview and evaluation by product experts)	Instead of it being knowledge engineers, who are to represent product information, to a greater extent this can be done by the product experts themselves
Structuring product information according to the chosen configurator software	The modelling restrictions of Product Model Manager and the mapping to the configurator knowledge basically eliminate this task
Implementing information in conceptual models into the configurator software	The automatic transfer of models in Product Model Manager to the configurator knowledge base eliminates this task
Facilitating tests of the configurator	Since product experts themselves can transfer their models into the configurator, there may not be a need for facilitation of tests

Table 1. Benefits from using the PMM-CSM when developing configurators.

4.3 Impacts on configurator maintenance

There are two distinct ways of handling the maintenance phase of configurator projects, which may be combined: 1) update external models and transfer the changes to the configurator, and 2) update the configurator and describe the changes in the external documentation. For the first approach, i.e. defining changes in external documentation before implementing them in the configurator, this corresponds to the process illustrated in Figure 7, and the associated advantages correspond to what is described in Table 1. On the other hand, if the procedure is to make changes in the configurator and subsequently describing them in external

documentation, another set of benefits can be achieved from the use of the PMM-CSM (Figure 8), as described in Table 2.

Knowledge engineer/software expert task	**Possible benefits from using PMM and the PMM-CSM**
Identify the gap between existing external documentation and the configurator	The PMM-CSM automates this task by comparing the PMM file to the configurator file and displaying the differences
Translate changes in the configurator to a language understood by product experts	The PMM-CSM automates this task by using its mapping between the configurator language and PVMs/CRC-cards in PMM
Update external representations when changes of the configurator occur	The PMM-CSM automates this task by translating changes of the configurator to updates of relevant models in PMM

Table 2. Benefits from using the PMM-CSM for documentation of configurators.

As seen in Table 2, the three major tasks of making external documentation of what is implemented in a configurator can all be automated and thus the time and resources used for these tasks can be significantly reduced or even eliminated. Furthermore, the chance of errors when transferring information between a configurator and conceptual models is minimised.

5 Case study

This section describes the company in focus, their motivation for using PMM, and two projects at this company in which the features of the PMM-CSM were used.

5.1 Case description

The use of the PMM-CSM was studied at the Danish company Novenco. Novenco is among the world leaders in supplying HVAC+R systems. Novenco develops and manufactures heating, ventilation, air-conditioning and refrigeration solutions for land and marine applications, and also offers a range fire fighting products. Founded in Denmark in 1947, Novenco is currently represented in 9 countries and employs about 650 people. With support from the regional Danish trade development project 'User Driven Innovation in Value Chains', in April 2008, Novenco initiated a project of standardising parts of their product assortment and making their product information easier to communicate internally and to external parties, i.e. more formalised and explicit representations. The experience with the PMM-CSM was obtained through using it in this project.

5.2 Phase 1: Getting acquainted with PMM

In the project in focus, the first major assignment was to create an overview of the existing solution space for air handling units for the cargo segment and to optimise it. The main part of this project lasted from April to July 2008, with some minor revisions to follow. The overview of the existing solution space was created in PVM models, drawn by using MS Excel. At the beginning of July, the same process was to be repeated for the air handling units for the land segment with the support of another knowledge engineer. Based on arguments of being able to create PVM models significantly faster if using PMM instead of Excel, Novenco decided to switch to PMM.

During the first part of the project, in which Excel was used for creating PVM models, PVM models were almost solely produced by the original knowledge engineer in the project. However, after having acquired PMM, soon persons from the product design and engineering departments began creating PVM models of various product types by using PMM. Three Novenco representatives from the software department, the engineering department and the product development department respectively were interviewed about the transition from Excel to PMM. According to them, compared to the use of Excel, PMM decreases the time for drawing work needed when creating PVM models by at least 50 percent. Also, these employees consider that the time needed for learning the proper use of PVMs and the possibilities of making incorrect PVM models can be significantly reduced by using PMM, because PMM ensures that models elements obey the defined notation formalism and can only be combined in valid ways.

In the beginning, PMM was solely used as a modelling tool, implying manual transfers of information from PMM models to configurators. However, after some time, the IT department of Novenco began investigating the possibility of automating these transfers. At this point of time a beta-version of the PMM-CSM had just been completed. Novenco decided to invest in a version of this software, customized to suit their particular configurator software and ways of using it. The first two phases of using the PMM-CSM included: (1) import of data from existing configurator into PMM; and (2) export of PMM models to a configurator followed by continuous synchronisation between the PMM models and the configurator models. These phases are described in the two subsequent sections.

5.3 Import from configurator into PMM

Novenco offers a particular type of engineered product that consists of 5 subtypes/families, which each consists of solution spaces of thousands of possible instances. To support the sales department (ensure that only legal product designs are sold) some years ago a configurator for these products was developed by using a standard configurator software shell. Unfortunately, in this configurator some important rules were missing and it had not been updated for some time, for which

reason its outputs were questionable. This meant that sales persons often made great efforts to avoid using this configurator. Furthermore, the standard configurator software used for developing this configurator differed from the software used to create the other configurators of Novenco. Thus, Novenco was very keen on replacing their old configurator with a new and more correct configurator, created in the same type of configurator software as used for their other configurators. The process for carrying out this project was as follows:

- The PMM-CSM was used for importing the knowledge base of the old configurator into PMM.
- Modelling sessions including the knowledge engineer and relevant product experts were held around the PVM models in PMM, displayed by using a projector. The product experts discussed the PVM models, which were continuously refined until the necessary agreements had been made.
- The completed PVM models were handed over to the IT department.
- At the time of this process, the customisations of the PMM-CSM were ongoing, for which reason the transfer of information into the configurator software shell was done manually.

The alternative to the first step of the described process would have been to study the knowledge base of the old configurator and, based on this, describe its contents manually. By using the PMM-CSM, this task was carried out automatically, i.e. a dramatic reduction of workload from hours or days to minutes. However, this task was only to be carried out once, for which reason the use of the PMM-CSM did not have a significant impact on the resources used at Novenco in the long run. On the other hand, had this been a task to be carried out frequently, this would have had a significant effect on the time and resources used at Novenco. In fact, this is what Novenco is currently doing for some of their configurators, namely using the PMM-SCM for the creation of external documentation of what has been implemented in a configurator each time the configurator is changed. These exports from configurators to PMM have been conducted without problems.

5.4 Export from PMM to configurator

Novenco has a subsidiary who shares the IT department with Novenco. At this subsidiary, possible benefits of using a configurator during the product design phase were identified. With an expectation that the use of the PMM-CSM could significantly reduce the amount of resources needed for the creation of a configurator, this project was initiated. The configurator software shell in focus was Configit v. 4.1. The aim of the project was to create a configurator that based on relevant design choices, could produce bills of materials. These bills of

materials are to be imported into the ERP system in order to attach prices. The project included the creation of three PVM models (for three distinct product families). For the first model, part of the needed product information was modelled by the product experts themselves, using PMM. These parts were combined with other existing documentation in a new PMM model by the knowledge engineer. The PVM model was discussed and refined in 4 sessions of 3 to 4 hours. The final PVM model in PMM was exported to Configit, and without any adjustments in Configit, the model could be compiled and a web-based runtime application could be generated. Using this runtime application, upon making approximately 50 selections, a bill of materials is generated (the number of selections varies dependent of particular choices).

Compared to the existing approach of transferring PVM models, created in software such as MS Visio or Excel, into a configurator, significant time savings were achieved by the use of the PMM-CSM. The task of transferring PVM models to a configurator knowledge base is a task that normally lasts days, weeks or months, depending on the model complexity and how well the conceptual model is mapped to the configurator language. However, by using the PMM-CSM in the current project, this task was done in a few minutes. Another significant benefit of using the PMM-CSM was identified during the modelling sessions. Often when creating large conceptual models of what to implement in a configurator, it can be difficult to predict exactly how such models would behave once they are implemented. However, the PMM-CSM allowed instant transfer to the configurator software so that these could be compiled and then tested in runtime during the modelling sessions. This procedure was frequently applied, which in many cases led to the discovery of incorrect or missing rules, for which reason the PVM models were altered. Thus, a later test phase including much communication concerning model adjustments between the product experts and the IT department seems to have been avoided.

Figure 9 shows a small part of the PVM model from the project and Figure 10 shows a screenshot from the web-based runtime model, which was automatically generated.

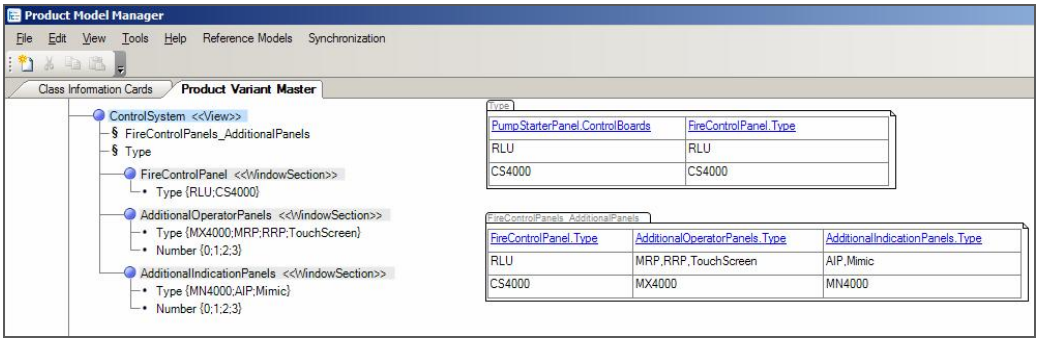

Figure 9. Part of the PVM model in PMM.

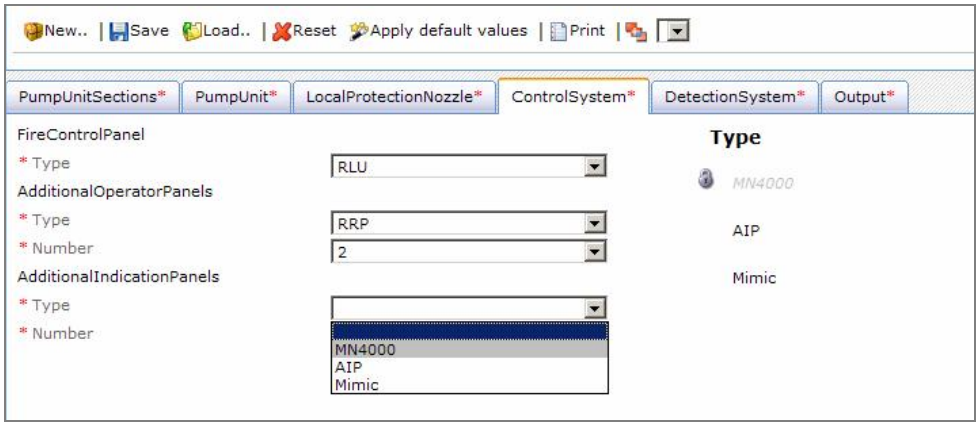

Figure 10. Part of the runtime interface.

Although the automatically generated runtime application can perform the basic functionality needed, some additional developments were in progress at the time of the study. More specifically, the generated web-pages had to be modified to improve the layout, and a script that generates a text file for import into the ERP system had to be made. Such developments are traditional programming tasks, which must be handled outside the PMM and Configit software.

6 Conclusions

This paper described the emergence of a software module (referred to as the PMM-CSM) that connects the conceptual modelling tool, PMM, to standard configurator software shells (at the moment two software types). The paper deduced how software as the PMM-CSM can impact the ways in which configurators are developed and maintained. More specifically, five new use patterns, made possible by the PMM-CSM, were described. The paper deduced the possible effects from applying these use patterns compared to exiting ways of developing and maintaining configurators. During the development phase, the

most remarkable benefit of using the PMM-CSM is that it enables conceptual models created in PMM to be automatically transferred to configurator software, after which these in principle can be compiled and launched instantly. In the maintenance phase of configurator projects, the PMM-CSM can automate the tasks of identifying gaps between external documentation and the configurator knowledge base and translate the changes made in the configurator software to PVM and CRC-card models in PMM.

Having defined the potential benefits of using the PMM-CSM, a study of its use at Novenco was carried out. In this case, the PMM-CSM was used for converting multiple models created in two types of configurator software into PVM models in PMM. These conversions were done in a matter of minutes, which should be seen in relation to the fact that such tasks normally lasts hours or days when done manually. Furthermore, the case showed that a relatively complex PVM model created in PMM could be translated automatically into a configurator knowledge base, in which it could be compiled and a web interface generated. Thus, it has been shown that a PMM based PVM model can be converted automatically into a software configurator, which in principle is ready for use. This can be done in a few minutes, compared to being something that normally lasts weeks or months. Another, key lesson was that the PMM-CSM enabled instant creation of configurable software prototypes during the modelling sessions. These configurable software prototypes seemed to give the product experts a much better understanding of the task at hand, for which reason this may have reduced the time used on the modelling sessions and implied that many misunderstandings were avoided.

In summation, this paper has described a new software module for automatic synchronization of information in conceptual models and configurator software shells. Based on the successful application in two projects, the technology seems to hold great promises to ease tasks of developing and maintain configurators, although more studies are obviously needed for more solid confirmation.

References

Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Petrone, G., Schäfer, R. and Zanker, M. (2003). "A framework for the development of personalized, distributed web-based configuration systems", *AI Magazine*, vol. 24, no. 3, 93-108.

Argyris, C., Putnam, R. and Smith, D.M. (1985). *Action science*, Jossey-Bass, San Francisco, CA.

Edwards, K., Hvam, L., Pedersen, J.L., Møldrup, M. and Møller, N. (2005). *Udvikling og implementering af konfigureringssystemer: økonomi, teknologi og organisation*. Final report from PETO research project. Department of Manufacturing Engineering and Management, Technical University of Denmark, Lyngby, Denmark.

Felfernig, A., Friedrich, G.E. and Jannach, D. (2000). "UML as domain specific language for the construction of knowledge-based configuration systems", *International Journal of Software Engineering and Knowledge Engineering*, vol. 10, no. 4, 449-469.

Forza, C. and Salvador, F. (2002). "Managing for variety in the order acquisition and fulfilment process: The contribution of product configuration systems", *International Journal of Production Economics*, vol. 76, no. 1, 87-98.

Forza, C. and Salvador, F. (2007). *Product information management for mass customization*. Palgrave MacMillan, New York.

Fowler, M. (2005). *UML distilled: A brief guide to the standard object modeling language (3rd edition)*. Addison-Wesley, Boston, MA.

Hansen, B.L. (2003). *Development of Industrial Variant Specification Systems*. PhD thesis. Department of Manufacturing Engineering and Management, Technical University of Denmark, Lyngby, Denmark.

Harlou U. (2006). *Developing product families based on architectures - Contribution to a theory of product families*. PhD thesis. Department of Mechanical Engineering, Technical University of Denmark, Lyngby, Denmark.

Haug, A. (2007). *Representation of Industrial knowledge - as a basis for developing and maintaining product configurators.* PhD thesis. Department of Industrial Management and Engineering, Technical University of Denmark, Lyngby, Denmark.

Haug, A. and Hvam, L. (2007a). "A comparative study of two graphical notations for the development of product configuration systems", *International Journal of Industrial Engineering*, vol. 14, no. 2, 107-116.

Haug, A. and Hvam, L. (2007b). "Product Structured Class Diagrams to support the development of Product Configuration Systems", in W.J. Mitchell, F.T. Piller, M. Tseng, R. Chin, B.L. McClanahan (eds.) *Proceedings of the MCPC 2007*, MIT, Boston, MA.

Haug, A. and Hvam, L. (2007c). "The modelling techniques of a documentation system that supports the development and maintenance of product configuration systems", *International Journal of Mass Customisation*, vol. 2, no. 1/2, 1-18.

Haug, A. and Hvam, L. (2009). "CRC-cards to support development and maintenance of product configuration systems", *International Journal of Mass Customisation*, vol. 3, no. 1, 38-57.

Haug, A., Degn, A., Poulsen, B., and Hvam, L. (2007). "A prototype of a documentation system that supports the development and maintenance of product configuration systems", *WSEAS Transactions on Information science and Applications*, vol. 4, no. 5, 1048-1055.

Haug, A., Lisbjerg, T. and Hvam, L. (2008). "A software tool for design and documentation of configurator knowledge bases", in K. Edwards, L. Hvam, T Blecker, F. Salvador, G. Friedrich (eds.) *Proceedings of the PETO/IMCM'08 Joint Conference*, Technical University of Denmark, Lyngby, Denmark.

Haug, A., Lisbjerg, T. and Hvam, L. (2009). "A Software System for the Management of Generic Product Information Models", in J. Malmqvist, G. Gustafsson (eds.) *NordPLM'09: Proceedings of the 2nd Nordic Conference on Product Lifecycle Management*, Chalmers University of Technology, Göteborg, Sweden.

Hvam L., Pape S., Jensen K.L. and Riis, J. (2005). "Development and maintenance of product configuration systems: Requirements for a documentation tool", *International Journal of Industrial Engineering - Theory Applications and Practice*, vol. 12, no. 1, 79-88.

Hvam, L. (2004). "A multi-perspective approach for the design of Product Configuration Systems – an evaluation of industry applications", in *Proceedings of the International Conference of Economic, Technical and Organizational aspects of Product Configuration Systems*, pp. 13-25, Technical University of Denmark, Lyngby, Denmark.

Hvam, L., Mortensen, N.H. and Riis, J. (2008). *Product customization*. Springer-Verlag, Berlin and Heidelberg.

Hvam, L., Riis, J. and Malis, M. (2002). "A multi-perspective approach for the design of configuration systems", presented at *15th European Conference on Artificial Intelligence*, Lyon, France, July 21-26 2002.

Mortensen, N.H., Yu, B., Skovgaard, H. and Harlou, U. (2000). "Conceptual modeling of product families in configuration projects", in *Papers from the Workshop at ECAI , 14th European Conference on Artificial Intelligence*, pp. 68-73, Humboldt University, Berlin, Germany.

OMG (2005). *Unified Modeling Language: Superstructure: Version 2.0 (formal/05-07-04)*. OMG, Nedham, MA.

Sabin, D. and Weigel, R. (1998). "Product configuration frameworks - A survey", *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 4, 42-49.

Susman, G.I. and Evered, R.D. (1978). "An assessment of the scientific merits of action research", *Administrative Science Quarterly*, vol. 23, no. 4, 582-603.

Waterman, H., Tillen, D., Dickson, R. and De Koning, K. (2001). "Action research: a systematic review and guidance for assessment", *Health Technology Assessment*, vol. 5, no. 23, 1-166.